Sistemas distribuidos
Laboratorio #3

Ayudante: Maximiliano Pérez Rodríguez
Profesor: Daniel Wladdimiro Cottet

Primer semestre 2017

1 Objetivo

Implementación de un sistema híbrido con uso de caché para el comportamiento de datos entre *peers*.

2 Enunciado

Dentro de las redes P2P unas de las aplicaciones que existe, es realizar intercambio de información entre cada uno de los integrantes de la red. De esta manera, cada *peers* puede solicitar o emitir información a cualquier otro *peers* que esté dentro de la red. Si bien este tipo de aplicaciones son bastante populares, existe un alto nivel de complejidad en la implementación de éstas, dado el protocolo de comunicación. Sin ir más lejos, en una red P2P de tipo no estructurada realiza una búsqueda completa teóricamente no es posible, por lo que encontrar una información exacta en la red se realiza un proceso complejo y lento, teniendo que realizar distintas optimizaciones en la búsqueda de la consulta.

Debido al problema anterior, es que se propone una red que utilice las redes P2P de tipo híbrido para la implementación de un sistema de intercambio de información. En la Figura 1 se muestra la primera parte del diseño de la red, donde su *overlay* está conformado por conexiones aleatorias en base al nodo inicial, que sería el *peer* morado, que a su vez será un *super-peer*. De esta manera, tenemos una red P2P no-estructurada, con un *super-peer* en su interior.

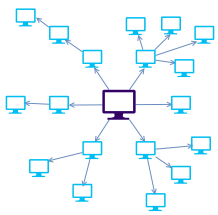

Figure 1: Red no estructurada.

Luego, la red para ampliarla, sin tener que seguir realizando conexiones aleatorias según la red no-estructurada mencionada anteriormente, se propone realizar un *overlay* determinístico en base a s *super-peers*. Cada uno de estos *super-peers* están inmersos en una red no-estructurada como se ejemplificó en la Figura 1. De esta manera, tenemos una arquitectura de una red P2P como se observa en la Figura 2, donde existe un conjunto de s *super-peers* que forman una red estructurada, y cada uno de ellos está en una red no-estructurada que posee un tamaño k_i de *peers*, donde k varía entre $[n - m]$ e i corresponde a la red no-estructurada.

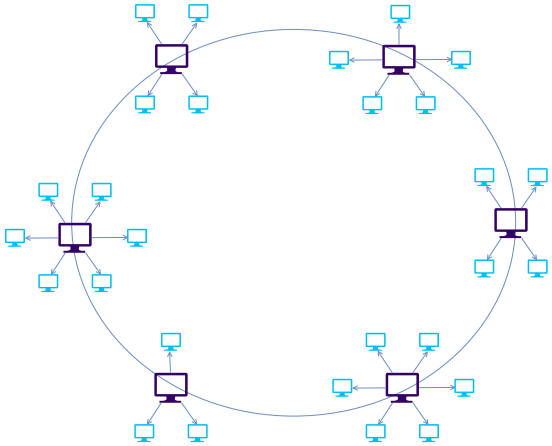

Figure 2: Red estructurada en base a los *super-peers*, los cuales están bajo una red no estructurada.

Como se había mencionado anteriormente, se debe realizar una red estructurada. Para esto, se debe utilizar el protocolo *Chord* para su confección. Cabe destacar que para efectos del laboratorio no es necesario aplicar las funciones de *join* ni *leave* en la red, sólo debe realizar el *lookup* para realizar la búsqueda de la consulta en el *overlay*. Esto significa que al inicializarse la red, cada *super-peer* debe asignarle un *id* mediante una función *hash*[1] a su IP y puerto del *super-peer*. Para efectos del laboratorio se debe crear una IP única, la cual es asignada de manera aleatoria, y los puertos disponibles rondan entre el $[3000 - 4000]$, el cual también es elegido de manera aleatoria. Una vez implementado el *overlay*, se debe inicializar el `DHT` de cada uno de los *super-peers*, donde éste se determina según el protocolo *Chord*.

Por otra parte, para realizar la búsqueda en la red, es necesario considerar dos variables. Una, un *id* del *super-peer* donde se va a aplicar la consulta posteriormente a su sub-red, y el *id* de la consulta realizada. Cabe destacar que puede ser que la consulta realizada no se encuentre, lo cual es completamente posible. Dado esto, se propone utilizar una base de datos para *peers* con tamaño d, donde los datos que posee cada *peer* no son únicos, por lo tanto, pueden repetirse en la red.

Prosiguiendo con la búsqueda de la consulta, el primero paso es encontrar al *super-peer* en la red no-estructurada, dado esto, se propone utilizar k *random-walk* paralelos con un TTL de tamaño t. En el caso que se localice el *super-peer*, los demás *random-walk* que todavía están buscando no deben repetir la consulta en el caso que terminen encontrando el *super-peer*. Luego, el *super-peer* realiza la consulta en base al protocolo *Chord* a la red, por lo tanto realiza una búsqueda según su `DHT`. Una vez identificado el *super-peer* de destino, este último realiza una consulta según *k-random-walk* (como al inicio de la búsqueda) a la red no-estructurada donde participa, y en caso de encontrar el resultado, este *super-peer* lo envía directamente al *super-peer* que hizo la búsqueda, para finalmente retorne al *peer* que solicitó la búsqueda. Para el retorno de

[1]Recomendación: Utilice una función *hash* ya implementada como SHA o MD5.

la búsqueda del *random-walk*, se propone (para optimizar la búsqueda) realizar un mensaje directo entre el *peer* inicial y el final.

Realicemos una consulta de ejemplo para ver el funcionamiento de la búsqueda de información. Supongamos que tenemos una red como se ve en la Figura 3, donde el *peer* rojo quiere realizar una búsqueda del documento *a* al *super-peer b*. De esta manera, el *peer* rojo realiza una consulta vía *k random-walk* hasta llegar al *super-peer*.

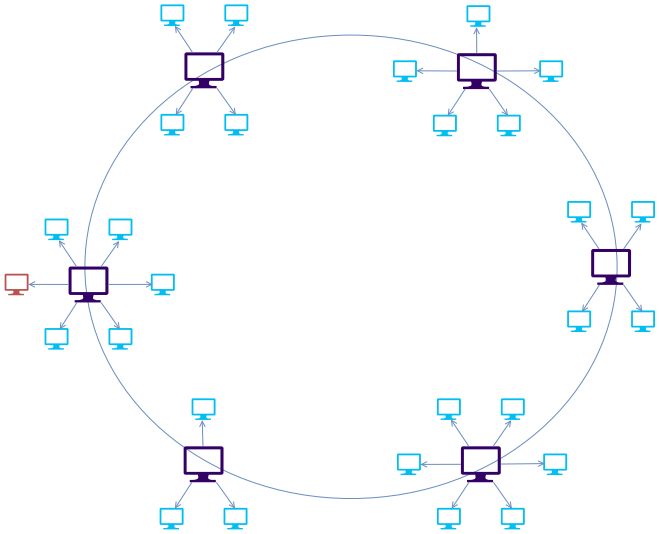

Figure 3: Búsqueda de una consulta en la red híbrida.

Una vez en el *super-peer*, éste analiza su DHT haciendo donde debe saltar como se ve en la Figura 4. Una vez localizado el *super-peer* de destino, se realizando los saltos correspondientes según el protocolo *Chord*, para posteriormente realizar búsqueda según un *k-random-walk* en la red no-estructurada donde participa este *super-peer*.

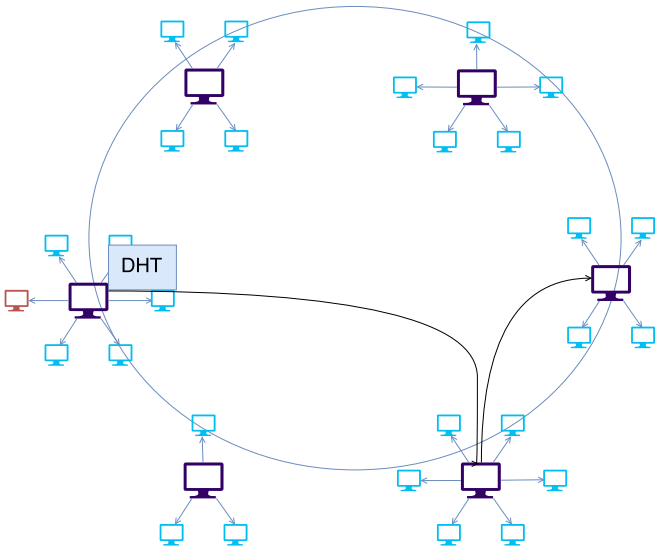

Figure 4: Búsqueda de una consulta en la red híbrida.

Recordemos que la búsqueda al ser localizada, ésta debe enviar de manera directa el mensaje entre los *peers* de inicio y destino, en cada una de las búsquedas. Ya sea por el *lookup* del protocolo *Chord*, como la búsqueda por *random walk*.

Una de las optimizaciones que se desea realizar al sistema, es utilizar un sistema de caché de tamaño c con una política `LRU` en cada *super-peer* como se muestra en la Figura 5. De esta manera, al realizar cada salto por la red estructurada, antes de consultar a la `DHT`, se propone consultar al caché, y en caso de encontrarse la respuesta, retornarla.

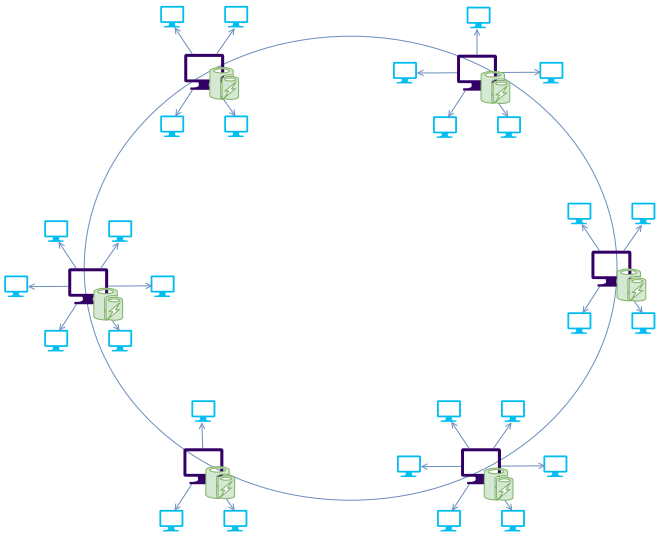

Figure 5: Implementación del caché a la red híbrida.

3 Evaluación

La entrega del laboratorio deberá ser subida al Moodle del curso en Usachvirtual. Este debe ser subido en un archivo comprimido `tar.gz`, con su respectivo archivo de configuración y código fuente para la ejecución del simulador.

Todo parámetro mencionado en el enunciado, debe ser ingresado por el archivo de configuración que provee el simulador Peersim.

En el caso que no se cumpla con ninguna de las solicitudes descritas anteriormente, se evaluará con la nota mínima.

La *fecha de término* del laboratorio es el **15 de Junio**.

4 Importante

- Detección de copias de trabajos (actuales o anteriores) se calificará con la nota mínima.
- Copy/Paste tendrá un descuento de 1 punto si no cuenta con el debido reconocimiento de la fuente desde donde se obtuvieron los extractos del código.
- El trabajo debe ser coherente con lo especificado en el enunciado.
- Por cada día de atraso desde la fecha de entrega del código tendrá un descuento de 1 punto.

5 Referencias

- PeerSim. http://peersim.sourceforge.net/
- Tanenbaum, A. & Van Steen, M. - Distributed Systems: Principles and Paradigms. *Cap. 3*
- Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Computer Communication Review, 31(4), 149-160. https://pdos.csail.mit.edu/papers/ton:chord/paper-ton.pdf